1 Introdução

Os processos tradicionais de desenvolvimento de software são baseados em paradigmas e no modelo mental herdados da revolução industrial. A forma analítica de encarar os problemas, dividindo-os em partes menores e depois atacando cada uma delas como problemas isolados, a forma de encarar o trabalhador como um recurso, substituível, todo esse modelo mental foi formado a partir do trabalho do Taylor [Taylor, 1911].

Em grandes projetos, um dos maiores desafios é manter-se no curso do planejamento, em termos de escopo, prazo e qualidade. De acordo com o modelo mental Taylorista, a forma mais simples de controlar atrasos ou acelerar o desenvolvimento de um projeto é adicionar mais recursos ao mesmo. O PMBOK reflete isso muito bem ao recomendar a técnica de *crashing*, ou seja, adicionar recursos ao projeto (humanos ou materiais) com o objetivo de acelerar seu progresso, mesmo que isso signifique a perda de eficiência. E como diz Brooks no livro *The Mythical Man-Month* isso é uma solução ineficaz, piorando a situação do projeto, *"adding manpower to a late software project makes it later"* [Brooks, 1975].

Considerando a forte associação do modelo mental Taylorista à revolução industrial, podemos observar que os reflexos nos processos, técnicas e métodos utilizados para gerenciar projetos hoje têm a sua base nas premissas de meados do século XX. Esses métodos funcionaram para produção de bens materiais, construções, indústrias, e foram adaptados para projetos de software. Entretanto, projetos de software, diferente dos modelos citados anteriormente, são mais dependentes de conhecimento empírico e comunicação, consequentemente, adicionar mais pessoas a esse meio só faz dificultar a eficácia da comunicação, por adicionar mais canais.

Em um processo ágil, utilizando XP ou Scrum, um time é formado seguindo uma sugestão de quantidade pequena de pessoas (7 ± 2). Mas, o que acontece quando existe a necessidade de aumentar a velocidade das entregas? O que acontece caso se tenha um projeto grande com um prazo de entrega curto? É possível utilizar Scrum em projetos de grande porte?

Analisando as possibilidades de generalizar as soluções em futuras aplicações, diante destes questionamentos, esta dissertação se propõe a examinar casos práticos em que foi utilizado Scrum em grandes projetos, enfatizando as dificuldades encontradas ao longo de todo o processo e as soluções adotadas, destacando as observações realizadas no ajuste e acompanhamento de projetos desenvolvidos em uma grande empresa sediada no Rio.

Uma das características do Scrum [Leffingwell, 2007], é o trabalho em pequenos times: *"Small, cross-functional teams work closely together in an open environment to produce incremental releases of a product in 30-day increments, or sprints"*.

Se Scrum prega a aplicação de equipes reduzidas, o que acontece com um grande projeto? Uma forma possível de "escalar" usando Scrum é criar múltiplos times. *"Many projects require more effort than a single Scrum Team can provide. In these circumstances, multiple Teams can be employed. Working in parallel..."* [Schwaber, K. 2004].

"Scrum is the first process with well-documented linear scalability. When you double team size in a well-implemented Scrum, you can double software output, even when the teams are distributed and outsourced". [Vaihansky, P. / Sutherland, J. / Victorov, A. 2006]. Essa afirmativa conflita com o livro *The Mythical Man-Month "adding manpower to a late software project makes it later"* [Brooks, 1975] que diz que adicionar recursos a um projeto em andamento fora do seu cronograma, fará com que esse projeto se atrase ainda mais, por questões de perdas devidas à comunicação entre os participantes.

Considerando um grupo de pessoas precisa manter uma comunicação diária entre si, ao aumentar o número de pessoas aumentam as combinações dessas

comunicações, fazendo com que, em situações patológicas, o esforço gasto em comunicação possa tornar-se maior do que o esforço gasto em produção. Uma maneira de diminuir o ruído nessas comunicações é o uso de algumas práticas no ambiente ágil, que não só facilitam a comunicação como também a detecção de problemas de forma mais rápida, que em consequência podem ser atacados mais rapidamente, seja na sua solução ou no replanejamento do plano original do projeto. Algumas dessas práticas que serão abordadas nessa dissertação são: integração contínua, entregas pequenas e frequentes, definição prévia do ciclo de desenvolvimento de componentes [Leffingwell, 2007].

1.1. Organização da dissertação

Essa dissertação está organizada da seguinte forma:

No capítulo 2 são apresentados alguns métodos ágeis, mais detalhadamente o Scrum.

Algumas boas práticas utilizadas em um ambiente ágil no que diz respeito à utilização dessas metodologias em um projeto grande são exemplificadas no capitulo 3.

O capitulo 4 irá abordar duas formas possíveis de organizar pessoas em um projeto grande, em um único time grande ou compondo diversos times pequenos. Outro ponto abordado são os casos de empresas que aplicaram Scrum em projetos de médio e grande porte e as lições que aprendidas a partir de suas experiências.

Por fim, no capítulo 5 são apresentadas as conclusões e propostas de trabalhos futuros.